Empowered by Wireless Communication: Distributed Methods for Self-Organizing Traffic Collectives

SÁNDOR P. FEKETE and CHRISTIANE SCHMIDT
Braunschweig University of Technology
and
AXEL WEGENER, HORST HELLBRÜCK and STEFAN FISCHER
University of Lübeck

In recent years, tremendous progress has been made in understanding the dynamics of vehicle traffic flow and traffic congestion by interpreting traffic as a multi-particle system. This helps to explain the onset and persistence of many undesired phenomena, e.g., traffic jams. It also reflects the apparent helplessness of drivers in traffic, who feel like passive particles that are pushed around by exterior forces; one of the crucial aspects is the inability to communicate and coordinate with other traffic participants.

We present distributed methods for solving these fundamental problems, employing modern wireless, ad-hoc, multi-hop networks. The underlying idea is to use these capabilities as the basis for self-organizing methods for coordinating data collection and processing, recognizing traffic phenomena, and changing their structure by coordinated behavior. The overall objective is a multi-level approach that reaches from protocols for local wireless communication, data dissemination, pattern recognition, over hierarchical structuring and coordinated behavior, all the way to large-scale traffic regulation.

In this article we describe three types of results: (i) self-organizing and distributed methods for maintaining and collecting data (using our concept of *Hovering Data Clouds*); (ii) adaptive data dissemination for traffic information systems; (iii) methods for self-recognition of traffic jams. We conclude by describing higher-level aspects of our work.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; E.1 [**Data Structures**]: Distributed Data Structures

General Terms: Algorithms; experimentation; theory

Additional Key Words and Phrases: Organic computing, traffic, traffic jams, self-organizing systems, pattern recognition, Hovering Data Clouds, Organic Information Complexes.

The work described in this article is based in parts on the conference papers [Fekete et al. 2006] and [Wegener et al. 2007]. It has been supported within the DFG Priority Programme "Organic Computing" (SPP 1183), project "AutoNomos", grant numbers FE 407/11-1, FE 407/11-2, FE 407/11-3, Fi 605/12-1, Fi 605/12-2, Fi 605/12-3.

Authors' addresses: S. P. Fekete and C. Schmidt, Department of Computer Science, Braunschweig University of Technology, Braunschweig, Germany, email:{s.fekete,c.schmidt}@tu-bs.de; A. Wegener, H. Hellbrück and S. Fischer, Institute of Telematics, University of Lübeck, Lübeck, Germany, email:{wegener,hellbrueck,fischer}@itm.uni-luebeck.de.

1. INTRODUCTION

1.1 Complex Systems and Organic Computing

A standard scientific method for *understanding* complicated situations is to analyze them in a top-down, hierarchical manner. This also works well for *organizing* a large variety of structures; that is why a similar approach has worked extremely well for employing computers in so many aspects of our life.

On the other hand, our world has grown to be increasingly complex. The resulting challenges have become so demanding that it is impossible to ignore that a large variety of systems has a very different structure: The stability and effectiveness of our modern political, social and economic structures relies on the fact that they are based on decentralized, distributed and self-organizing mechanisms. (In this context, see [Surowiecki 2004] for a recent non-fiction bestseller.)

Until not very long ago, scientific efforts for studying computing methodologies for decentralized complex systems have been very limited. Traditional computing systems are based on a centralized algorithmic paradigm: data is gathered, processed, and the result is administered by one central authority. Each of these aspects is subject to obstructions. On the other hand, "Living organisms [...] are to be treated as dynamic systems and contain all infrastructure necessary for their development, instead of depending on coupling to a separate thinking mind. We call this computing paradigm *organic computing* to emphasize both organic structure and complex, purposeful action" [Organic Computing Initiative 2004]. The importance of organic computing has been motivated as follows: "The advantages of self-organizing systems are obvious: They behave more like intelligent assistants, rather than a rigidly programmed slave. They are flexible, robust against (partial) failure, and are able to self-optimize. The cost of design decreases, because not every variant has to be programmed in advance" [Müller-Schloer et al. 2004].

Two important properties of organic computing systems are *self-organization* and *emergence*. We follow the definition of de Wolf [De Wolf and Holvoet 2005] who gives the following definitions:

> "Self-organization is a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control."

and

> "A system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel w.r.t. the individual parts of the system."

This work was conducted in the spirit of organic computing. In the following we will show how to combine aspects of distributed computing and communication (introduced in Section 1.2) with a new concept of algorithms and data structures, in order to deal with the challenge of influencing a very important complex system: traffic. Clearly, road traffic by itself (as introduced in Section 1.3) exhibits both properties postulated by de Wolf. Vehicles interact on the micro-level without external control by local interactions (self-organized) and due to this micro-level interactions structures at the macro-level arise, such as traffic jams or convoys.

1.2 Distributed Communication

A critical aspect for influencing a complex system, no matter whether in a centralized or decentralized manner, is making use of distributed communication, which is becoming more and more feasible by the advance of modern technology. Today's applications for distributed systems rely on unicast or multicast communication, where communication partners are identified by layer-3 addresses. Prominent and well-known examples are Client-Server-Applications like webbrowsing. When mobile ad-hoc networks (MANETs) first came up, research concentrated on efficient routing to keep up the existing communication paradigm. By increasing the number of nodes and introducing dynamics in the network due to node mobility or switching nodes on and off in a large multi-hop network, routing based on addresses became a hard challenge. In the meantime systems have grown larger, to the point where they consist of thousands of cheap battery-powered devices that organize themselves and open a new research field called *sensor networks*. Limitations of bandwidth, computing power and storage in sensor networks drive a new communication paradigm. While these systems need to be designed for frequent node failures, redundancy was introduced in the network, decreasing the importance of single nodes and addresses. The stringent separation of the lower communication layers has also become open for discussion, as it introduces computational and networking overhead especially when identification of nodes loses importance.

1.3 Traffic

Traffic is one of the most influential phenomena of civilization. On a small scale, it affects the daily life of billions of individuals; on a larger scale, it determines the operating conditions of all industrialized economies; and on the global scale, traffic has a tremendous impact on the living conditions on our planet, both in a positive and in a negative way.

All this makes traffic one of the most important complex systems of our modern world. It has several levels of complexity, reaching from individual actions of the drivers, over local phenomena like density fluctuations and traffic jams, traffic participants' choice of transport mode and time, regional and temporal traffic patterns, all the way up to long-range traffic development and regulation.

In recent years, tremendous progress has been made in understanding the dynamics of traffic flow and traffic congestion. Arguably the most significant contribution has come from physics, interpreting traffic as a multi-particle system. These models explain how the complexity of traffic emerges from the self-organization of individuals that follow simple rules. They also meet the popular appeal of systems of interacting, autonomous agents as problem-solving devices and models of complex behavior. In short, the "decentralized" view, which goes beyond attempts at centralized simulation and control, has improved our understanding of traffic.

1.4 Combining Communication and Mobility

Modern hardware has advanced to the point that it is technically possible to enable communication and coordination between traffic participants. At this point, this is mostly seen as a mere technical gadget to facilitate driving, but the far-reaching, large-scale consequences have yet to be explored. Making use of the technical

possibilities of communication and coordination should allow significant changes in the large-scale behavior of traffic as a complex network phenomenon. However, combining mobility and communication for coordinated behavior does not only solve problems, it also creates new ones, as it is a challenge in itself to maintain the involved ad-hoc networks, as well as the related information that is independent of individual vehicles.

To the best of our knowledge, the idea of combining the above aspects, i.e., self-organizing traffic by combining ad-hoc networks with distributed decentralized algorithms has not been studied so far. This may be because it requires combining a number of different aspects, each of which has only been developed by itself in recent years: mobile ad-hoc networks, models for large systems of self-driven multi-particle systems, as well as algorithmic aspects of decentralized distributed computing, possibly with elements of game-theoretic interaction.

1.5 Our Work

Our basic idea is to develop a decentralized method for traffic analysis and control, based on a bottom-up, multilevel approach. Beyond the motivations described above, it should be stressed that an aspect of particular relevance is *scalability:* while the computational effort for a centralized approach increases prohibitively with the number of vehicles, a decentralized method relies on neighborhood interaction of constant size.

At this point, we focus on a number of different aspects:

(1) One is the key concept of *Hovering Data Clouds* (HDCs), which are virtual structures that exist independent of particular carriers.

(2) Data dissemination in large-scale ad-hoc networks formed by moving vehicles in a complex traffic situation requires adaptive protocols for traffic information systems.

(3) The self-recognition of traffic situations by distributed reasoning, and without a central operating center allows drivers to benefit from joint identification of traffic jam types and parameters.

(4) Based on these insights and structures, we can aim at actually *changing* the behavior of traffic.

The rest of this paper is organized as follows. In the next Section 2, we give a broad survey of related work, subdivided into distributed communication and data dissemination (2.1), traffic and telematics (2.2), and traffic as a self-organizing system (2.3). In Section 3, we describe AutoCast, our approach for adaptive data dissemination. Section 4 introduces the concept of Hovering Data Clouds (HDCs), and describes how they can be used for distributed recognition of traffic jams. Extensions for identifying types and parameters are described in Section 5, along with simulation results. Further extension of our ongoing work is discussed in the concluding Section 6.

2. RELATED WORK

2.1 Communication Networks and Data Dissemination

In today's communication world, wireless networks such as GSM in the wide area and WLAN in the local area range have become ubiquitous. Still, most applications using these networks rely on a thoroughly managed infrastructure such as, for instance, base stations in GSM or access points in WLAN. Many research activities, however, already go one step further and make use of the fact that more and more mobile devices with radio communication capabilities are available. These devices are not necessarily bound to communication infrastructures, but can instead create a spontaneous network, a so-called ad-hoc network, in order to allow communication among some of the processors (and not necessarily to the outside). In its sophisticated form, some of the processors in an ad-hoc network act as relay stations and transport data from a source to a destination that is not in direct radio range (multihop ad-hoc network). For an overview on ad-hoc networks see the book by Perkins [Perkins 2001].

Unicast is the most popular way of communication and can be seen as the standard for Client-Server based communication in the internet. Nowadays, using wireless multi-hop networks many applications have different requirements and various protocols and communication approaches have been developed in the past to match new applications' demands.

In sensor networks the communication paradigm shifts away from the node-centric way where data is delivered between nodes identified by addresses to a data-centric way of communication. The basic idea of the data-centric communication is that nodes subscribe to a type of data identified by a unique name and receive data associated with this name as shown in [Intanagonwiwat et al. 2003; Ratnasamy et al. 2001; Shenker et al. 2003]. Since data is often sent to one or only few sinks in sensor networks, approaches like [Ye et al. 2002] deal with moving sinks while the rest of the network stays immobile. In any case, routes between the originator of data and its subscribers are needed to transport data through the multi-hop network. All these approaches fail for the fast changing network topologies in vehicular ad-hoc networks (VANETs).

Projects like [Franz et al. 2005] address the new challenges of VANETs but often use data dissemination approaches limited to emergency data. Like other examples as [Xu et al. 2004; Oh et al. 2006] emergency notifications are assumed to occur only rarely, statically and will be short-lived. By contrast, our approach is able to handle numerous data units in parallel, even when they are disseminated at the same time to arbitrary directions and created at arbitrary positions in the network.

Approaches that concentrate on disseminating traffic conditions, like [Wischhof et al. 2003; Xu and Barth 2006], focus on the adaptation of broadcast interval, e.g., according to the vehicle's speed. The proposed techniques are closely bound to specific applications with fixed sized road segments and distinguish only between regular communication and emergency data. In particular, they are not designed for dynamic appearance and heterogeneity of data units with individual life time and suddenly occurring long-lived data that describes, e.g., a traffic jam.

A further optimization to save bandwidth while ensuring that every node gets as much data as possible is described in [Heinzelman et al. 1999]. Each data unit is

represented by a hash value. In a unicast approach, new data is sent after a three way handshake comprising advertisement, request, and delivery of data units.

Each approach has its individual optimum working condition and is mainly created on the fly to solve a particular problem. In Section 3.1 we will derive step by step a generic optimized protocol for data dissemination inspired by our AutoNomos application.

2.2 Traffic and Telematics

As the interest in guiding and organizing traffic has been growing over recent years, the scientific interest in traffic as a research topic has developed quite dramatically. For an overview ("Traffic and related self-driven many-particle systems"), see the excellent survey [Helbing 2001]. Obviously, research on traffic as a whole is an area far too wide for a brief description in this short overview; we focus on a particular strain of research that is particularly relevant for our proposed work, as it appears to be most suited for simulation and extension to decentralized, self-organizing systems of many vehicles.

It is remarkable that until the early '90s, efforts for simulating traffic were based on complex multi-parameter models of individual vehicles, resulting in complex systems of differential equations, with the hope of extending those into simulations for traffic as a whole. Obvious deficiencies of this kind of approach are manifold:

(1) Because the behavior of even just an individual vehicle is guided by all sorts of factors influencing a driver, the hope for a closed and full description appears hopeless.

(2) Determining the necessary data for setting up a simulation for a relevant scenario is virtually impossible.

(3) Running such a simulation quickly hits a wall; even with today's computing power, simulating a traffic jam with a few thousand individual vehicles based on such a model is far beyond reach.

A breakthrough was reached when physicists started to use a different kind of approach: instead of modeling vehicles with ever-increasing numbers of hidden parameters, try to consider them as systems of many particles, each governed by a very basic set of rules. As Nagel and Schreckenberg managed to show [Nagel and Schreckenberg 1992], even a simple model based on cellular automata can produce fractal-like structures of spontaneous traffic jams, i.e., complex, self-organizing phenomena. Over the years, these models [Nagel 1995] were generalized to two-lane highway traffic [Rickert et al. 1996], extended for simulating commuter traffic in a large city [Rickert and Nagel 1997], and have grown [Nagel 2002] to the point of being used for real-time traffic forecasts for the 2250 km of public highways in the German federal state of North Rhine-Westphalia [Kaumann et al. 2000; Pottmeier et al. 2003] (see www.autobahn.nrw.de.) Also, see the book chapter by Nagel [Nagel 2003].

A closely related line of research uses an approach that is even closer to particle physics; see [Nagel et al. 2003] for an excellent overview of models for traffic flow and traffic jams, with about 150 relevant references. Among many others, particularly remarkable is the approach by [Krauß 1997]: this model reproduces properties of

phase transitions in traffic flow, focusing on the influence of parameters describing typical acceleration and deceleration capabilities of vehicles. This is based on the assumption that the capabilities of drivers to communicate and coordinate are basically restricted to avoid collisions, which until now is frustratingly close to what drivers can do when stuck in dense traffic.

Parallel to the scientific developments described above, the interest in and the methods for obtaining accurate traffic data has continued to grow. The employment of induction loops and traffic cameras has been around for quite a while, but despite of enormous investments, e.g., 200 Mio. Euros by the German Federal Ministry for Transport, Building and Urban Affairs (BMVBW) [Bundesministerium für Verkehr, Bau und Stadtentwicklung 2002] for putting up systems for influencing traffic, the limits on tracking individual vehicles, as well as following particular traffic substructures are obvious. A more recent development is the use of *floating car data:* By keeping track of the movements of a suitable subset of vehicles (e.g., taxis in Berlin city traffic), the hope is to get a more accurate overall image of traffic situations, both in time and space [Kwella and Lehmann 2000]. However, even this approach relies on the use of the central processor paradigm, and does not allow the use of ad-hoc networks for the active and direct interaction and coordination between vehicles.

2.3 Traffic as a Self-Organizing Organic System

The structure of traffic is a phenomenon that is self-organizing at several levels; see [Gershenson and Heylighen 2002] for a philosophical discussion of self-organization in multi-level models. But even though the behavior of and the interaction between motorists has been observed for a long time, the possibilities arising from modern technology allowing direct and decentralized complex interaction between vehicles has hardly been studied. The only efforts we are aware of combine game theory with traffic simulations. (For example, see the symposium organized by traffic physicist Schreckenberg with game-theory Nobel laureate Selten [Schreckenberg and Selten 2004].) Neither make use of mobile ad-hoc networks and distributed algorithms in large networks.

Several research projects focus on enhancing traffic flow by distributed methods. A model based on multi-agents systems is described in [Bazzan and Junges 2006]. Depending on local congestion, a dynamic toll is charged to influence the drivers' routing decisions. Camurri et al. [Camurri et al. 2006] propose a distributed approach using Co-Fields for routing vehicles around crowded areas on a urban grid-like road map. Techniques for detecting traffic anomalies are also developed in centralized systems like the "System for Automatic Monitoring of Traffic" as proposed in [Bandini et al. 2002]: Video cameras are installed along highways and traffic events are derived and monitored over time based on the composed view of the cameras.

All these approaches offer methods that try to solve certain problems arising in the field of traffic management. Our approach offers a self-organizing framework for decentralized applications; depending on the setting (urban, highway) and the current traffic situation, different strategies can be integrated to detect and overcome adverse traffic conditions.

3. DATA DISSEMINATION

3.1 Protocol

The proposed protocol *AutoCast* aims at applications that need to communicate in a many-to-many manner, without a need to set up an association or connections between network nodes. In a traffic information system, each car contributes to the knowledge of road conditions that may be important for nearby cars.

In general, dissemination of data in mobile ad-hoc networks can be achieved in two ways. This may either be by the movement of network nodes (see [Grossglauser and Tse 2001]) or by multi-hop ad-hoc communication between nodes. Because communication is much faster than carrying data piggybacked on moving nodes, it is preferred in most scenarios. However, node movement can support communication when networks are partitioned, e.g., by using opposite-lane traffic for bridging gaps between cars.

Because ad-hoc networks already use a broadcast medium, unicast communication is an artificial constraint. Sending broadcast messages is more efficient than unicast messages, gaining even more with increasing network density. Even if a particular data unit is not useful for a node, it can assist in further dissemination of the data unit.

The most intuitive technique is pure *flooding*, where each node receiving a data unit rebroadcasts it exactly once and as soon as possible. Flooding can be fast for fully connected networks. However, the single-rebroadcast property causes network partitions to stop data forever. As a consequence, flooding cannot cross partitions in the network; additionally, it jams the wireless channel in dense networks with a broadcast storm [Ni et al. 1999].

A well-established method for disseminating data slowly and more reliably, even when network partitions occur frequently, is a periodic rebroadcast of received data with a short delay. As described in [Hellbrück and Fischer 2004], the protocol *MILE* is designed for the exchange of location information. We enhance *MILE* to work with generic data units instead of location information. By randomly choosing several data units from all locally known data units when broadcasting, data dissemination reaches an acceptable speed. The main drawback is that this technique does not scale with increasing network density and increasing number of data units in the network.

It can easily be seen from the detailed results in Section 3.3 that both basic approaches have advantages and disadvantages.

In order to measure the best possible performance, we introduce a *theoretical* protocol as benchmark. This protocol assumes unlimited transmission rate, propagation speed of light, and a perfect intuition of the sender as to which data units need to be sent to which nodes, just in the moment when they are able to receive them correctly. This happens magically, especially when network partitions merge again without any delay and additional communication overhead.

As a first improvement, we optimize *MILE* by reducing the amount of data that needs to be transmitted periodically. The idea is to use simple and well-known hash values. Nodes create short hash values from data units, so-called IDs—sometimes also called metadata—and send these instead of complete data units. This complete list of IDs is broadcast periodically by each node, together with a subset of data

units. If a node gets an incomplete list of IDs from a neighbor, it will add the missing data units in its next update packet. By this simple extension we avoid an explicit request of missing data by individual nodes and thereby achieve an additional reduction of bandwidth usage. The drawback is an increased delay, as nodes add the content of the data units only when other nodes within the transmission radius are found that do not know about a particular data unit. We call this extension *MILE on-demand*.

We combine the ideas of *flooding* and *MILE on-demand* for further reducing the communication overhead and increasing the speed of data dissemination, as well as the data delivery ratio. We call the new protocol *AutoCast*; it works as follows, making use of two basic mechanisms.

Newly generated data units are flooded through the network in the beginning, but only a portion of the nodes participate actively in the flooding. Instead of using the magic numbers of 60% to 80% as a forwarding probability (as suggested in [Haas et al. 2001]), we adapt to the dynamics and irregularity of the network. Nodes derive the probability from their number of neighbors. To avoid broadcast storms, on average only two nodes of those receiving a new data unit rebroadcast it. It has been shown in [Hellbrück and Fischer 2002] that on the average only about 40 % of the neighboring nodes receive the data unit for the first time as 60 % of the nodes have received the previous broadcast already. Consequently, a node with 10 neighbors forwards a data unit with a probability of $2/(10 \cdot 0.4) = 0.5$, which is according to the results of [Haas et al. 2001] for this scenario. However, the forwarding probability for single nodes will decrease further when network density increases, thus ensuring scalability. In a traffic jam, the number of neighbors can reach 100 cars easily where with our approach an individual node forwards data unit with a probability of 0.05.

The second mechanism was introduced by *MILE on-demand*: Periodically rebroadcast IDs of elder data units, because due to bad luck, flooding might stop sometimes when several nodes do not forward data. Periodic rebroadcasts are also important to reach locally consistent states in the network, especially when new nodes join the network or network partitions merge. Like the forwarding probability, the rebroadcast interval also depends on the number of neighbors, and in addition on the network dynamics. In a static network in which nodes neither move nor appear, periodic rebroadcasting does not help at all, as flooding already delivered the data units to all reachable nodes. On the other hand, increasing speed of nodes combined with frequent network partitioning will force the update interval to reach zero, which means all nodes broadcast permanently. Thus, the key issue is to determine the optimal update interval; furthermore, how can individual nodes calculate it on their own?

Assume that we know n as the size of a node's one-hop neighborhood. The waiting time until the next rebroadcast is calculated as n/p_{ref}, where p_{ref} is a constant that describes the desired number of broadcasts per second. We will explain our choice of p_{ref} more detailed in Section 3.3. A positive side-effect is the following: Cars driving near a network partition boundary send twice the number of packets as cars driving inside that partition, as border nodes have only half the expected number of neighbors.

Table I. Average neighborhood sizes for different penetration rates.

penetration rate [%]	5	10	20	30	40	50	60	70	80	90	100
neighborhood size	0.9	1.8	3.6	5.4	7.2	9	10.8	12.6	14.4	16.2	18

3.2 Simulation Setup

After having discussed five different approaches (including the *AutoCast* protocol), we set up a simulation environment to evaluate and compare them.

We have chosen a dynamic highway scenario with varying network density and the influence of opposite-lane traffic for our protocol's performance. Cars drive on a highway section of 10 km, with two lanes in each direction and an average speed of 100 km/h. In order to reach realistic node movements that will appear in VANETs due to individual cars' behavior, we used the traffic simulator SUMO (see [Krajzewicz et al. 2002]), which is based on the microscopic car following model described in [Krauß 1997]. The mean distance between two consecutive cars on one lane is around 110 meters, leading to an overall mean car density of 36 cars/km. Because road density is hard to compare to other simulation setups, Table I shows neighborhood sizes in our setup that result from different fractions of cars equipped with AutoNomos devices, so-called *penetration rates*.

The nominal duration of our traffic simulation is 26 min, with an initial startup time of 10 min to spread the cars all over the road. The last 16 min of the generated cars' mobility are stored into ns2-trace files, each with a different penetration rate.

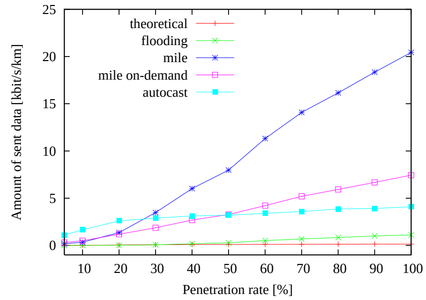
ns-2 [Fall and Varadhan 2001] is used as network simulator for performance evaluation of the different data dissemination protocols. All simulations use standard IEEE 802.11 MAC-layer, with a radio range of 250 m and a bandwidth of 1 Mbps in combination with the Two-Ray Ground propagation model. Periodically, the car driving closest to km 5 at the appointed time generates a data unit, which is disseminated over the simulated road (5 km in each direction); the unit's lifetime is set to 50 seconds.

Each protocol is simulated with different penetration rates, as shown in Table I, and between two and 50 data units that need to be disseminated concurrently.

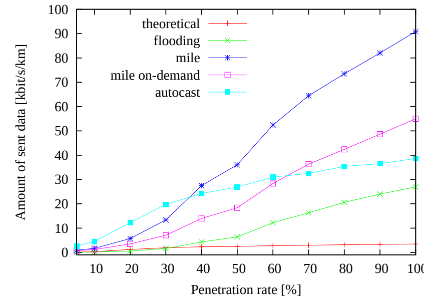
3.3 Results

Figure 1 shows the results of the simulation, with each line in the graph showing a protocol. In all plots the x-axis shows the penetration rate of cars that participate in the VANET. The left column shows the protocols' behavior, if only two data units are disseminated. Figures on the right show the results for 50 concurrent data units. In order to leave enough network capacity for other applications and protocols, data dissemination should be optimized for low bandwidth; Figures 1(a) and 1(b) show the transmitted data per km, as concurrent communication is possible if sending nodes have a distance of more than four times the transmission radius. Figures 1(c) and 1(d) show the achieved speed of data dissemination. A fast speed is preferable, because data units may comprise time-critical data like emergency messages. To evaluate the success of data dissemination, Figures 1(e) and 1(f) present the amount of successfully delivered data units.

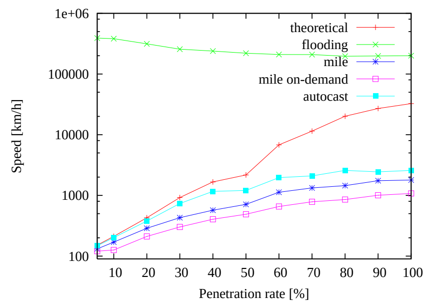
The *theoretical* protocol sends a broadcast only if it will successfully inform a car, so less than 4 kbit/s/km of bandwidth are consumed in any case. With low



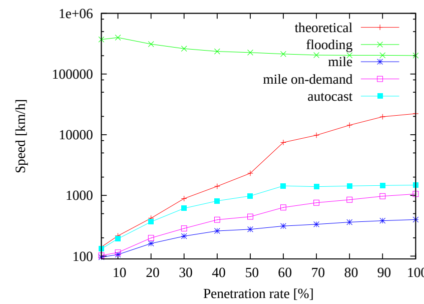
(a) Radio channel usage per km, 2 simultaneous data units



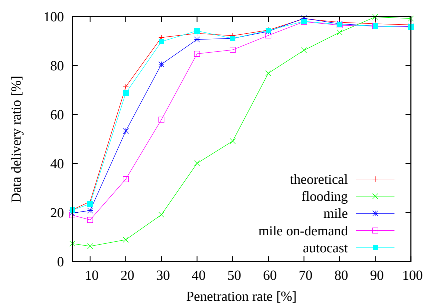
(b) Radio channel usage per km, 50 simultaneous data units



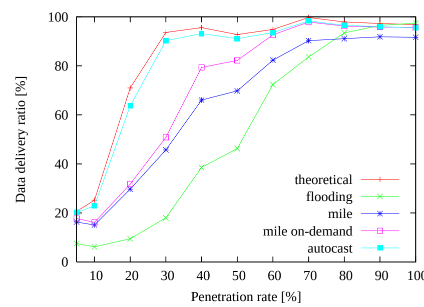
(c) Speed of data traffic, 2 simultaneous data units



(d) Speed of data traffic, 50 simultaneous data units



(e) Data delivery ratio, 2 simultaneous data units



(f) Data delivery ratio, 50 simultaneous data units

Fig. 1. Simulation results comparing the different data dissemination algorithms for few data units (left column) and 50 data units (right column).

penetration rate, the bounding factor for data speed is almost completely the cars' driving speed. As expected, it rises with increasing penetration rate, up to more than 20000 km/h. This speed cannot be achieved by any other protocol, as in reality there is a trade-off between data speed, data delivery ratio and rebroadcasting interval. The data delivery ratio shows that a reasonable usefulness can be achieved with a minimal penetration rate of 30 % standing for 10.8 equipped cars per km. With a further increase in the number of cars, the ratio of delivered data units grows only marginally.

At first sight the *flooding* protocol performs surprisingly well. It consumes few bandwidth and achieves a speed of above 100,000 km/h. However, the poor data delivery ratio puts that result in the right perspective, as with pure flooding a data unit will stay in its network partition. Consequently, flooded data is delivered either very fast or never.

With regard to enhancing the data delivery ratio, in particular in the case of low penetration rates, the *MILE* protocol achieves a remarkable improvement, approaching the theoretical results. If more data units need to be disseminated than what fits into one broadcast packet, the achieved data speed decreases from nearly 1800 km/h to under 400 km/h, even in case of a 100 % penetration rate. Moreover, the data delivery ratio drops as well.

Due to the exchange of data unit IDs, the protocol *MILE on-demand* can suppress the rebroadcasting of full data units that are already known by cars in the direct vicinity. The drawback of this method is a slight decrease in data speed, because the sender needs to know about missing data units before delivering them. Due to this effect *MILE on-demand* performs worse than pure *MILE* in case of only few data units. Nevertheless, the protocol's performance remains stable if more data units need to be handled. So far all protocols use fixed broadcast intervals of 2s. This results in a linear increase of bandwidth usage when more cars participate in the VANET.

As mentioned in Section 3.1, *AutoCast* produces a constant number of broadcast packets per second (p_{ref}), no matter how many nodes generate them. In order to calculate p_{ref} for our scenario, we analyze the *MILE on-demand* curve and find a minimum of 60 % penetration rate for a data delivery ratio above 90%. With 10.8 neighbors, i.e., $p_{\mathrm{ref}} = 10.8$ cars in the neighborhood per 2 s, about 5 packets/s are transmitted. The value of p_{ref} is a good choice for our scenario, but is definitely not the optimum for all ad-hoc networks. This parameter needs to be analyzed in more detail; we will address this problem in future work. Nevertheless, bandwidth consumption remains stable, independent of the network density, and depending only on the number of concurrent data units. The data speed reaches about 2000 km/h, enough to cross Germany in less than 30 min. The data delivery ratio gets close to the *theoretical* protocol, so even the primary goal of reaching as many cars as possible is achieved.

AutoCast clearly outperforms the other protocols and gets close to the theoretical maximum with respect to data dissemination speed and data delivery ratio. Due to a limited network overhead, it leaves enough room for additional applications and protocols in the ad-hoc network.

4. HOVERING DATA CLOUDS FOR TRAFFIC JAMS

4.1 Hovering Data Clouds

A dynamically changing system such as a traffic jam consists of many ever-changing objects, as cars located at different positions keep moving with respect to back and front of the queue. If we want to maintain useful information related to the back of a traffic jam, we have to keep shifting the related roles from one car to the next, together with all relevant data. Thus, we look for a *local* data structure with the following properties:

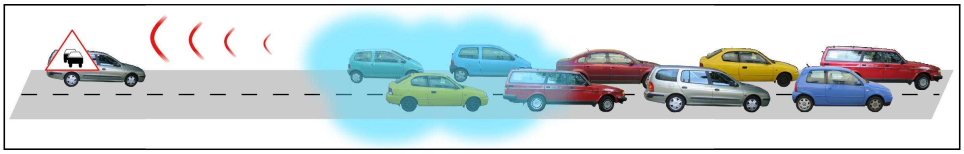


Fig. 2. A Hovering Data Cloud at the back of a traffic jam informs incoming vehicles.

—The data structure self-organizes with the onset of a traffic jam, and it ceases to exist when the jam disappears.
—It is located at a useful virtual location, which is defined by the traffic jam, e.g., its back.
—The structure continues to exist, even as their current carriers move or change their role.
—It contains up-to-date information that describes the traffic jam.

We call such a structure a *Hovering Data Cloud* (HDC). In this section, we describe how to deal with the above properties in the context of a relatively simple traffic scenario. Even though this presentation focuses on the context of traffic jams, there are a number of other scenarios that give rise to HDCs:

—Keeping track of large swarms of moving animals poses similar challenges of dynamically updating information, without relying on a fixed host.
—Pedestrian traffic also consists of a large number of individuals, each capable of (and nowadays routinely) carrying a mobile device. The resulting patterns can be even more complicated than highway traffic; see [Helbing et al. 2005] for an introduction. Quite clearly, HDCs can be used for a number of different pedestrian scenarios and applications.
—Even for scenarios with devices at fixed locations, HDCs may be useful: When a mobile object is tracked by a sensor field, historic information is accumulated over time that is not attached to the object (as it may not be equipped with an electronic device), but also not present at each individual sensor. Passing this information from host to host along the tracking path amounts to maintaining an HDC.

Figure 2 depicts a traffic scenario in which an HDC is already located at the end of a traffic jam. Vehicles approaching the HDC, participate in storing, and enrich the data by monitoring the underlying phenomenon. Furthermore, the data of the HDC is propagated and sent to oncoming vehicles as warning messages. Obviously, there are two reasons for communication between nodes in this example: firstly, vehicles holding the HDC communicate in order to maintain the HDC (intra-HDC communication); secondly, vehicles outside the HDC's area need to get informed about the HDC (inter-HDC communication).

We started our work on this concept by considering stationary HDCs [Wegener et al. 2006]; these arise when performing measurements at predefined locations. Using the simulator SUMO (Simulation of Urban MObility) [Krajzewicz et al. 2002] for generating mobility traces, and ns2 [Fall and Varadhan 2001] for network simulation, we were able to give a good reproduction of actual traffic 5 km away from the

HDC. However, the main interest and justification for HDCs arises in more dynamic situations, in which HDCs may not remain stationary. This work is described in detail in the rest of this section.

4.2 Scenario

We consider a single-lane highway. For convenience, we assume that cars move from left to right, as shown in Figure 3, i.e., positions with lower coordinates are shown on the left. Any vehicle carries a computing and wireless communication device, each with a unique identifier, and has a reliable way of measuring time and location, e.g., by using GPS. Cars can communicate if they are within broadcast range of each other; this range is denoted by R. Communication delays are relatively small, we assume 10 ms as transmission delay.

Now we examine traffic patterns. Depending on speed and traffic density, a traffic jam may form. This gives rise to two HDCs, one at the front and and one at the back of the queue, cf. Figure 3; these HDCs are maintained while the jam continues to exist. In the following we describe the details of how this can be achieved; note that the same basic variables and processes are maintained in each processor.

4.3 Variables

We will make use of the following variables and parameters for each processor.

$status_i$ (with possible values idle, joining, active) describes the current state of a processor, where the index $i \in \{\text{back}, \text{front}\}$ refers to the HDCs marking back and front of the traffic jam; initially, all processors are idle. In the absence of a traffic jam, idle processors near an HDC become active immediately. If a jam exists, processors become joining if they are within reach of the current HDC, described by a radius r_{HDC} around an HDC's current location. An active processor becomes idle, if it ceases to be near the boundary of the jam, i.e., if its distance to an HDC position exceeds r_{HDC}. Note that the status joining is only necessary in the presence of information that is not known to all processors.

state describes the values of all variables stored on a processor.

The coordinates $location_{\text{back}}$ and $location_{\text{front}}$ describe the current positions of an HDCs that mark back and front of the traffic jam. For clarity we should mention that the variables *loc, v* and ID refer to the actual processor, *l, g* and *ident* refer to the processor from which the actual message m was received. *buffer* is used for storing arriving messages. *clock* is used for keeping track of real time. *env* is a table for storing the data of all broadcasting processors within distance R (maximum size: $3 \times \lceil 2R/(\text{minimum distance})\rceil$).

If a processor has a left neighbor (i.e., a following car) within congestion radius *CR*, the flag p is set to 1. Analogously, $q = 1$ indicates a right neighbor (i.e., a preceding car) within congestion radius *CR*.

The auxiliary variables *ahead, congestion_counter, congestion_participant, congestion_participant_before, s, q_before, p_before, P_v, Q_v, p_v, q_v, t, front, back_id, front_id* are all initialized with 0; *back* is initialized with the largest possible value of a location on the road.

If status joining is necessary, then $v0_{\text{back}}$, $v0_{\text{front}}$ are the initial states of an HDCs.

t_{data}, t_{smdata}, $t_{\text{information}}$, $t_{\text{aheadInfo}}$ are times that are used for updating the indexed variables. t_{data}, t_{smdata} occur alternately. $t_{\text{information}}$, $t_{\text{aheadInfo}}$ describe times

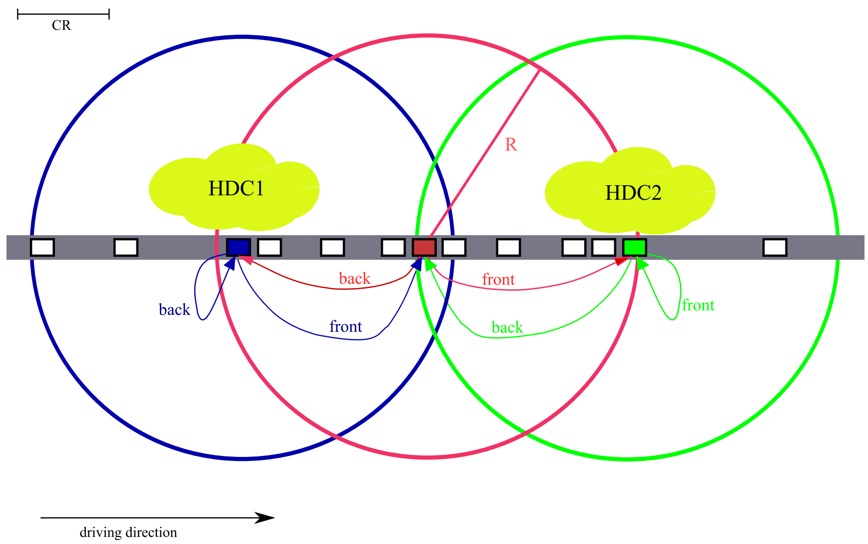


Fig. 3. Example for the two HDCs: different cars (rectangles) determine front and back (assumption: all drive sufficiently slowly), HDCs arise if all further conditions are met.

with evenly spaced intervals. The function settimer is used with two arguments: the first one indicating a certain point in time, the second one the callback function f which is evoked at this point in time (with *onTimer(f)*). A possibility to describe a point in time is to use the function next-multiple: Given a variable that indicates the time divided into intervals of equal length, next-multiple gives the next point in time where such an interval starts. Consequently, using the function settimer with the argument next-multiple(t_x) ($x \in \{\text{information, aheadInfo}\}$) sets the timer on the next time of t_x (even if some time elapsed since the last timer was set), cf. Figure 4. t_{ab} is the time period that passes until all processors within $2R$ have started their *Data* interval.

A parameter that is related to the presence of a traffic jam is the *congestion radius CR* that is a function of the current speed of vehicles; if two cars violate this critical distance, it may be an indication of a traffic jam.

Conversely, the *congestion velocity CV*, considers the velocity in relation to the current distance; if two cars move more slowly than appropriate for their current distance, this may also indicate a traffic jam. (Note that this second parameter filters out cars that tailgate at high speed. As pointed out by [Brilon et al. 2005], it works best to consider speed for recognizing traffic jams.)

Finally, d is the critical bound on the latency in *LBcast*.

4.4 Algorithm Description

We give details of the algorithmic steps; see Algorithm 1 (Appendix A).

We consider a discrete sequence of time slots, $t_i, t_{i+1}, \ldots$. The interval between two consecutive time slots is divided into two subintervals: a small interval (*smData*), and a bigger one (*Data*), cf. Figure 4. Thus, at the end of *smData (onTimer(smData))* the timer for *Data* is initiated and vice versa.

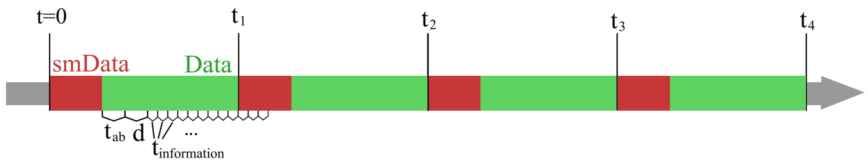


Fig. 4. Example for time slots and timer.

In *onTimer(smData)* the current processor position and its velocity are only broadcast within *CR*. If a processor receives such a message, it is treated in subroutine *onTimer(NewMessage)* (as all received messages)—after an additional delay of d (see *LBrecv(m)*). In case a processor receives such a message from ahead, we set p equal to 1. Analogously, a message from behind results in setting q equal to 1.

For sending more data (*position, velocity*, p, q) in a wider range in *onTimer(Data)* we distinguish several situations. Only if the processor is participating in an HDC or if it is neither caught up in a traffic jam (t_i) nor was so before (t_{i-1}) but falling below a certain velocity, it will send such a message. This enables us to reduce the amount of transmitted messages. However, non-active processors inside of a traffic jam or with sufficient high speed do not send, the information is either not important yet or the processor is not a potential participant of a traffic jam.

Describing the consequences of being a congestion participant, we need to consider how a processor achieves this situation. A processor receives messages of type *Data* from its surrounding processors within R. The data of each such processor is stored in *env*. Afterwards, it is checked whether the sending processor is located close (less than *CR*) to the receiver and if the velocity is sufficiently low, e.g., clearly below 60 km/h (see [Brilon et al. 2005; Krauß 1997].) If so, the back of the congestion is computed from the position of the two processors and the previous back position. Furthermore, the processor becomes a participant of the traffic jam. Similarly we check for all processors in *env* whether they are located close to the sending processor and fall below a velocity of *CV*. In both cases we increment a counter for the congestion (indicating a positive number of vehicles).

If the counter was incremented, the processor is close (less than r_{HDC}) to the back of the congestion, the back position has no left neighbor (thus, it is really the back) and all messages from processors within the range of R were received, then *Congestion* is invoked. The front of the jam is treated analogously; *CongestionAhead* is invoked here. If an HDC at the back has yet to receive information from an HDC at the front, then the position of *front* is set to the position of the most advanced processor within R.

The status of a processor is maintained in *Congestion*: if it is active, we set a timer for *Information*, i.e., as long as the processor is active, the position of an HDC at the back of the jam, the position of an HDC at the front, and the current speed of the back are broadcast. Only if the position of the back HDC has a value greater than the current processor location, the processor continues to broadcast, and processors approaching this position become *joining*.

In *CongestionAhead*, $status_{\text{front}}$ is updated; if the processor is active, the timer for *AheadInfo* is set. This means that the position of the front HDC is broadcast regularly, as long as the processor is active. When such a message *hdcdistance* is

processed, the back HDC variable *front* indicating the position of the front HDC is updated for an active processor: inactive processors between front and back HDC pass on the message towards the back.

With increasing distance, clustering and updating data is performed by the HDC transportation layer.

Thus, messages are broadcast to:

—relate the positions of the processors (messages broadcast in *onTimer(smData), onTimer(Data)*, processed in *onTimer(NewMessage)*),

—transmit the information of the HDC at the front of the traffic jam to the one at the back of the congestion (messages in *onTimer(AheadInfo)*),

—transmit the information of the back HDC to the following cars (messages in *onTimer(Information)*).

4.5 Further Aspects of HDCs

There are various extensions and generalizations of the above scenario. An obvious next step is to extend our ideas to two-lane highways (possibly stretching over several exits and even highway crossings), or more refined HDCs that reflect substructures in a traffic jam; other extensions and variants include the recognition of bottlenecks in traffic, e.g., caused by a convoy of slow trucks, accidents or emergency vehicles.

A qualitatively more challenging step is required when considering more advanced structures that consist of several HDCs: an HDC simply marks front or back of a traffic jam, but eventually we are interested in more complex interaction between all involved vehicles, e.g., when trying to smooth out complex stop-and-go patterns, mark advisable exits for following cars, or even map possible detours. We call such high-level structures *Organic Information Complexes* (OICs). The underlying idea is presented in Section 7. Details will be pursued and discussed in future work.

4.6 Simulation

We have implemented our method, using the traffic simulator SUMO (Simulation of Urban MObility, [Krajzewicz et al. 2002]) for generating traces of vehicular movement, and our own large-scale distributed network simulator Shawn [Kröller et al. 2005] for handling the algorithmic side. Enhanced by an OpenSG-based 3D visualization plug-in for Shawn by Baumgartner [Baumgartner 2006], the resulting simulation can produce videos; see Figure 5 for some screen shots. The initial state was chosen such that traffic jams emerge. Starting with these trace files, SUMO controls the movement of the vehicles (according to its internal rules, e.g., using the car-following model of [Krauß 1997]). The location and velocity of all cars are written in an input file for Shawn. Thus, Shawn only updates the location and speed and its focus is on the management of messages. In Figure 5 the resulting HDC at the front of the traffic jam is marked in brown (with corresponding cars labeled in yellow), while the HDC at the end is marked in red (as are the corresponding cars.) This shows that our methods do indeed produce the desired results.

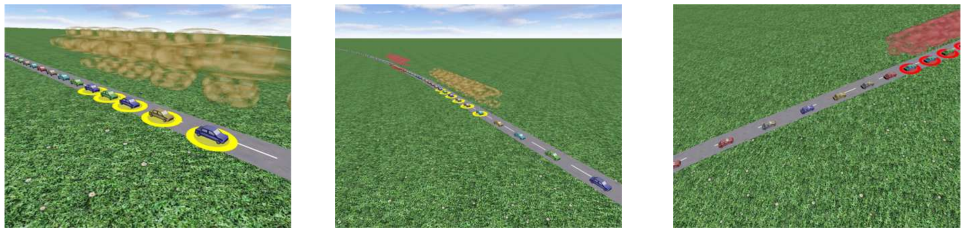


Fig. 5. Screenshots from the simulation of HDCs for recognizing a traffic jam. Left: front of the traffic jam. Center: overview with front and end. Right: end of the traffic jam.

5. DISTINGUISHING TYPES OF TRAFFIC JAMS

5.1 Traffic Jams

What exactly constitutes a traffic jam? According to [Krauß 1997] a "connected structure of vehicles, traveling at a velocity below a given threshold v_{thresh} will be called a jam if this structure contains at least one stopped vehicle." A good estimate is $v_{\mathrm{thresh}} = \frac{v_{\mathrm{max}}}{2}$, v_{max} being the maximum velocity.

However, as every driver knows from his or her own experience, not all traffic jams are alike; in particular, some appear to be an act of nature (e.g., getting stuck in a blocked highway after an accident), while others seem to have appeared out of thin air. When trying to improve traffic flow and energy consumption of vehicles, recognizing these distinctions is of crucial importance, as it constitutes the prerequisite for developing successful strategies.

5.2 Types of Traffic Jams

Another line of research investigates the characteristic properties of congested traffic and tries to identify different congestion patterns. According to current research [Schönhof and Helbing 2007], there are five different basic types of traffic jams, which we will present here in detail as we will use this categorization:

—Pinned Localized Cluster (PLC):
Neither the front or back of the congestion move upstream or downstream, i.e., a traffic jam of fixed length is situated upstream of a bottleneck. (If there is oscillation, this an oscillating pinned localized cluster, OPLC). A congestion of this type may arise spontaneously, or by upstream traveling perturbation, which stops at the bottleneck. If the density increases, this can change into a different type of congestion, in particular into spatially extended congestion patterns.

—Oscillating Congested Traffic (OCT)):
This is a spatially extended congestion pattern, i.e., one that grows in length, meaning that the upstream end moves in space. An OCT may be caused by a perturbation or supercritical traffic density; once the bottleneck has been removed, the congestion gradually disappears. Characteristic for this type of congestion is that a driver passing through it experiences "more or less regular oscillations of speed", commonly known as stop-and-go traffic. (However, this is different from the technical definition below.) Frequency and amplitude of the oscillations are relatively stable, and oscillations themselves move at about 15 km/h in the

upstream direction. OCT is surrounded by free traffic.

—Stop-and-Go Waves (SGW):
They are closely related to OCT; they have a large characteristic amplitude, but no characteristic wave length. They have been observed for a long time, the first description we are aware of appears in [Edie and Foote 1958]. SGWs consist of a sequence of congestions, separated by free traffic; individual congestions are stable in length, so they are spatially confined. An average duration of a wave appears to lie between 4 and 20 minutes, their typical speed is about 15 km/h in the upstream direction. SGW arise either from small perturbations, or from PLCs; in most cases, they turn into free-flowing traffic, but also into PLCs.

—Homogeneous Congested Traffic (HCT):
In HCT, speed is very low and relatively homogeneous over a larger stretch of road. HCT tends to form a spatially extended congestion pattern, with the downstream front fixed (usually at a bottleneck), while the upstream front moves further upstream. When the bottleneck is removed, the downstream front moves upstream at a typical speed of 15 km/h.

—Moving Localized Cluster (MLC):
While SGW consist of several congestions, an individual one is called a MLC. Both upstream and downstream front move at about 15 km/h upstream, with limited spatial dimensions. A MLC is caused by a perturbation.

5.3 Distinguishing Congestion Types

Any driver can decide that he is in a traffic jam by simply looking out of the window. But what kind of congestion is it? This is a first important step for actually changing the traffic situation by allowing the right decisions to be made.

In Figure 6 it is shown how this can be achieved within our framework. In case a traffic jam emerges, this is observed and Hovering Data Clouds are established at the back and front of the jam. Considering the movement of these clouds over the time, as well as the speed of the cars between these HDCs enables us to distinguish the congestion types: A Pinned Localized Cluster is characterized by a fixed upstream end, i.e., in case the back HDC stays at a fixed location we may identify a PLC. In the event of a Hovering Data Cloud at the back moving upstream, we consider the velocity profile of the cars between the front and the back HDC: oscillations of speed indicate an Oscillating Congested Traffic. To identify Homogeneous Congested Traffic, it is now sufficient to look at the distance between the front HDC and the back HDC. If this distance increases over time, we may identify a spatially extended traffic jam; with the other criteria: an HCT. For the final distinction between Stop-and-Go Waves and a Moving Localized Cluster, it suffices to check whether there are any nearby clusters of similar amplitude. The lack of nearby clusters indicates a single isolated traffic jam, here a MLC. Hence, keeping track of the location of the HDCs, the velocity profile of the cars between and messages on nearby clusters of congested traffic allows us to differentiate the five congestion types.

6. IMPROVING THE FLOW OF TRAFFIC

The grand challenge for any kind of traffic research is to actually improve the flow of traffic, in terms of travel time, energy consumption, or a combination of both.

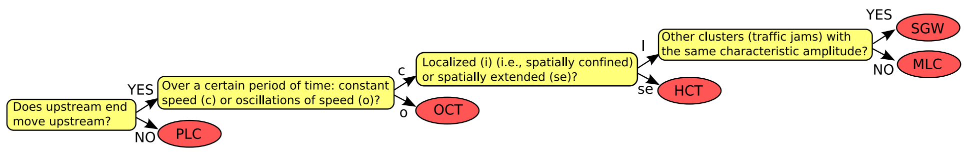


Fig. 6. Schematic distinction of traffic jam types. These considerations are carried out on the HDC level for each cluster.

Over the years, many different methods have been tried and even more have been suggested, but quite clearly, most have been too crude to have a real impact.

The methods for recognizing types of congestion are a first step in this direction; this has an impact both on the microscopic level, where making drivers aware of the type of traffic situation has an impact on useful behavior, as well as the macroscopic level, where routing is closely intertwined with forecasting, which is dependent not just on the amount of existing congestion, but also on its foreseeable development. In particular, there are a number of actions that can be taken at different levels:

—Identifying a pinned localized cluster comes with the clear recognition of a cause. This makes it possible to pinpoint possible action at the bottleneck, either by removing it, or by updating medium-range traffic information if removal is impossible; in the latter case, the identification of PLC means that drivers can be sure they can proceed swiftly once they are beyond the obstacle, which is not the case for stop-and-go waves.

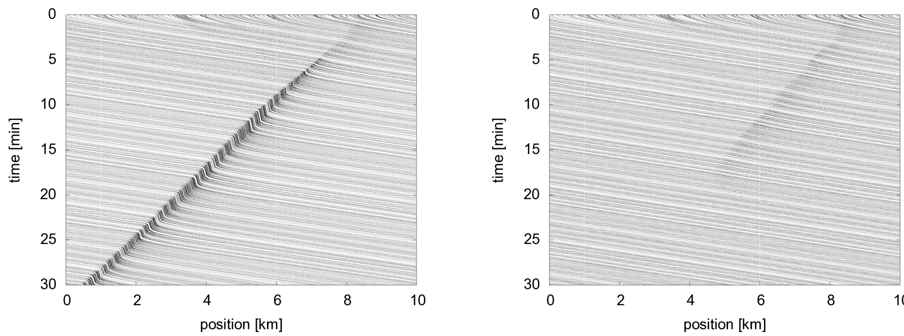
—While the cause for homogeneous congested traffic is similar to that of a pinned localized cluster, the effect of removing the bottleneck is different; furthermore, large-scale re-routing and forecasting are absolutely vital for reducing further flow into the jam, and thus prevent the congestion from causing even larger problems.

—The onset of oscillating traffic triggers a number of different actions. On a small scale, driver behavior can be influenced in order to improve flow; we have recently developed methods that work surprisingly well for this purpose, results are reported in a forthcoming paper [Fekete et al. 2010]. On a medium scale, increasing congestion may make it sensible to re-route traffic; this requires dealing with the large-scale optimization impact of flows over time (see [Skutella 2009] for a recent survey), i.e., turning to more advanced optimization methods, which are currently being studied in a separate context, see [Fekete et al. 2010].

—Identifying stop-and-go waves is particularly vital for preventing crashes, as well as reducing unnecessary fuel consumption. Moreover, distinguishing interior waves (where swift acceleration is wasteful and even hazardous) as opposed to the front of the jam (where outflow from the jam may be impeded by distrustful drivers) is critical for the overall flow rate.

—Identifying a moving localized cluster is particularly useful for regulating outflow, and thus overall flow, by letting drivers realize that there are no further oscillations ahead.

Another way to improvement is illustrated in the two parts of Figure 7. In both diagrams each line depicts the trajectory of a vehicle on the road: the faster the travel speed of the vehicle, the lower the incline of the line. As vehicles get slower,



(a) (b)

Fig. 7. Illustration of vehicle flow over time (traffic density $\rho = 30\,\mathrm{veh/km}$). Figure 7(a) shows a traffic jam that emerges from local disturbances, with a speed limit of 105 km/h. Limiting the maximum speed by just 5 km/h to 100 km/h will avoid the traffic jam, as depicted in Figure 7(b).

the gradient of the lines increases with time. Due to individual behavior of drivers, a traffic jam occurs in Figure 7(a) at an unpredictable time and position. Figure 7(b) shows a very similar scenario, in which the above jam does not occur. We have the same amount of cars and types, but reduce the global speed limit from 105 km/h to 100 km/h. The simulation confirms that small changes in the setup (like a slight reduction in the speed of cars) result in very different and unpredictable traffic structures.

Previous work on these issues suggests implementing a dynamic global speed limit to reduce the occurrence of traffic jams (see [Treiber and Helbing 2001]). Although the speed limit is globally assigned, and thereby a very coarse-grained measure, it suffices in our scenario to avoid the traffic jam.

A more sophisticated approach was described in [Kesting et al. 2005]: the authors propose a system called Adaptive Cruise Control (ACC) that "automatically accelerates or decelerates a vehicle to maintain a selected gap, to reach a desired velocity, or to prevent a rear-end collision." It is based on car-following methods (i.e., each vehicle adapts its behavior to distance and velocity of the car preceding it.) As the authors demonstrated in simulations, automated driving strategies help to improve capacity and stability of traffic flow, even if only a limited fraction of vehicles are equipped with ACC. However, ACC does not make use of higher-level communication and coordination capabilities, which is what we are aiming for.

We foresee a huge potential for quick and local reactions in a future traffic information system with only a small impact for the single car. In general, it is a difficult task to evaluate the critical thresholds at which traffic structures evolve (as is the case for the traffic jam in Figure 7), in particular in dynamically changing traffic densities, as occurring in real life. We believe that our framework allows going a step even beyond ACC, by actually using the communication capabilities for self-organized and coordinated behavior of a whole group of vehicles, instead of just setting global or local parameters.

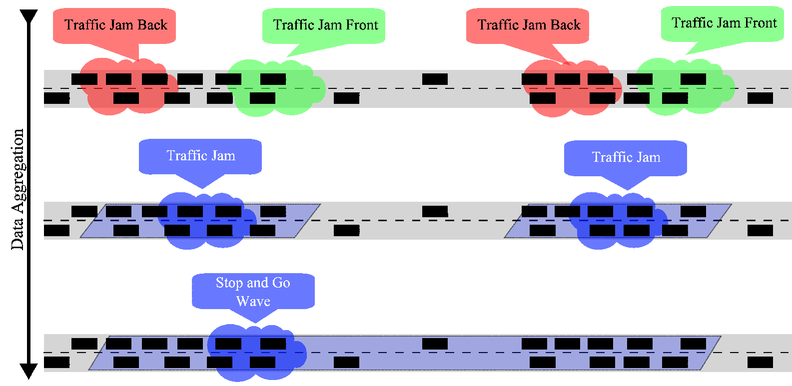


Fig. 8. Data aggregation by an Organic Information Complex to recognize a traffic jam.

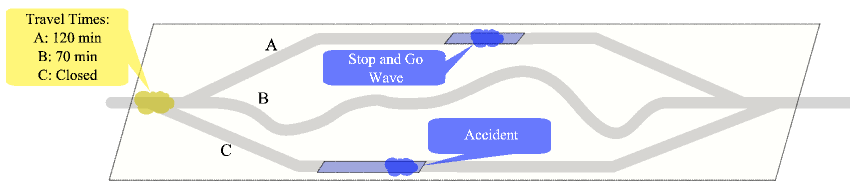


Fig. 9. An Organic Information Complex on a road network aggregates possible routes.

7. ORGANIC INFORMATION COMPLEXES

We conclude by sketching the next step beyond Hovering Data Clouds: using them to form larger-scale structures. We call these resulting structures *Organic Information Complexes* (OICs). While some of the resulting ideas are closely related to ideas described above, further aspects of OICs are subject to future work.

As described above, taking appropriate action may require finding means to describe higher-level, more complex structures, e.g., the individual stop-and-go waves in a SGW congestion. The data gathered by HDCs is raw data, e.g., describing the end of a traffic jam. We need to combine several HDCs to obtain higher-level information. As shown in Figure 8, HDCs resulting from the locally identified phenomena, e.g., *back of traffic jam* and *front of traffic jam* send out data units. When those data units accumulate, higher-level information can be built that describes the congestion from a higher perspective. When *front*- and *back*-HDCs gather a higher-level construct, an *Organic Information Complex* (OIC) arises. In our example the two *traffic jam*-OICs form a *stop and go wave*-OIC. This merging and aggregation of simple data into higher-value data is the basic idea for building up *Organic Information Complexes* (OICs).

A more complex scenario is depicted by Figure 9, where OICs have discovered congestions on roads A and C and react by sending this information towards incoming vehicles. When this information reaches intersections on the road network that can be utilized for bypassing the traffic jams, the different pieces of information can be turned into a journey-time prediction for the different routes.

Due to this in-network data aggregation, the overall system remains scalable. Note that HDCs and OICs are not necessarily bound to fixed locations, nor to specific network nodes. They arise in a self-organized manner, wherever the matching conditions occur. In almost the same manner, aggregation is not restricted to fixed locations, but happens as a result of appropriate concurring data units.

8. OUTLOOK

We have described a number of aspects of using wireless communication as the basis for self-organizing methods for participants in traffic. As demonstrated, this work extends from local protocols for communication all the way to high-level organization; methodically, it involves practical traffic models and simulations, as well as theoretical distributed algorithms. At this point, we have laid the basis for communication and coordination, and we have dealt with data dissemination; we were also able to demonstrate lower-level self-organization and self-recognition by making use of our concept of Hovering Data Clouds. First steps towards self-modification are on their way, and higher-level aspects are to follow.

Clearly, this is work in progress. We are optimistic that AutoNomos will continue to produce interesting results; even just realizing our ideas at an intermediate level may have far-reaching benefits for the way we act and interact in traffic.

REFERENCES

Bandini, S., Bogni, D., and Manzoni, S. 2002. Alarm correlation in traffic monitoring and control systems: A knowledge-based approach. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence*. Lyon, France, 638–642.

Baumgartner, T. 2006. Reliable and energy-efficient algorithms for sensor networks used in flood protection. M.S. thesis, Braunschweig University of Technology.

Bazzan, A. L. C. and Junges, R. 2006. Congestion tolls as utility alignment between agent and system optimum. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS '06)*. ACM, New York, NY, USA, 126–128.

Brilon, W., Regler, M., and Geistefeldt, J. 2005. Zufallscharakter der Kapazität von Autobahnen und praktische Konsequenzen. *Straßenverkehrstechnik 03 and 04*.

Bundesministerium für Verkehr, Bau und Stadtentwicklung. 2002. Programm zur Verkehrsbeeinflussung auf Bundesautobahnen 2002-2007. http://www.bmvbs.de.

Camurri, M., Mamei, M., and Zambonelli, F. 2006. Urban traffic control with co-fields. In *Proc. of the 3rd Int. Workshop on Environments for Multiagent Systems*. 11–25.

De Wolf, T. and Holvoet, T. 2005. Emergence versus self-organisation: Different concepts but promising when combined. *Engineering Self-Organising Systems 3464*, 1–15.

Edie, L. C. and Foote, R. S. 1958. Traffic flow in tunnels. In *Highway Research Board Proceedings*. Vol. 37. 334–344.

Fall, K. and Varadhan, K. 1989–2001. *The ns Manual*. The VINT Project - a collaboration between researchers.

Fekete, S. P., Köhler, E., Möhring, R. H., Nagel, K., and Skutella, M. 2007–2010. ADVEST: Adaptive traffic control. Project cluster funded by the German Federal Ministry of Education and Research.

Fekete, S. P., Schmidt, C., Wegener, A., and Fischer, S. 2006. Hovering data clouds for recognizing traffic jams. In *Proceedings of 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (IEEE-ISOLA)*. 213–218.

Fekete, S. P., Tessars, C., Hendriks, B., Schmidt, C., Wegener, A., Hellbrück, H., and Fischer, S. 2010. Improving the flow of congested traffic. *Manuscript*.

Franz, W., Hartenstein, H., and Mauve, M. 2005. *Inter-Vehicle-Communications Based on Ad Hoc Networking Principles - The FleetNet Project*. Universitätsverlag Karlsruhe.

Gershenson, C. and Heylighen, F. 2002. When can we call a system self-organizing? In *Advances in Artificial Life, 7th European Conference, ECAL 2003, Dortmund, Germany, September 14-17, 2003, Proceedings*, W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, Eds. Lecture Notes in Computer Science, vol. 2801. 606–614.

Grossglauser, M. and Tse, D. N. C. 2001. Mobility increases the capacity of ad-hoc wireless networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*. Anchorage, Alaska, 1360–1369.

Haas, Z., Halpern, J. Y., and Li, L. 2001. Gossip-based ad hoc routing. Tech. rep., Cornell University, Ithaca, NY, USA.

Heinzelman, W. R., Kulik, J., and Balakrishnan, H. 1999. Adaptive protocols for information dissemination in wireless sensor networks. In *5th Annual ACM/IEEE Conf. MobiCom '99*. ACM Press, New York, NY, USA, 174–185.

Helbing, D. 2001. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics 73*, 1067–1141.

Helbing, D., Buzna, L., Johansson, A., and Werner, T. 2005. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation Science 39,* 1, 1–24.

Hellbrück, H. and Fischer, S. 2002. Towards analysis and simulation of ad-hoc networks. In *Proc. 2002 Int. Conf. Wireless Networks (ICWN02)*. IEEE, Las Vegas, Nevada, USA, 69–75.

Hellbrück, H. and Fischer, S. 2004. MINE and MILE: improving connectivity in mobile ad-hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev. 8,* 4, 19–36.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F. 2003. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw. 11,* 1, 2–16.

Kaumann, O., Froese, K., Chrobok, R., Wahle, J., Neubert, L., and Schreckenberg, M. 2000. Online simulation of the freeway network of NRW. In *Traffic and Granular Flow '99*, D. Helbing, H. J. Herrmann, M. Schreckenberg, and D. E. Wolf, Eds. Springer, 351–356.

Kesting, A., Treiber, M., Schönhof, M., Kranke, F., and Helbing, D. 2005. Jam-avoiding adaptive cruise control (ACC) and its impact on traffic dynamics. In *Traffic and Granular Flow 2005*. Springer, 633–643.

Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. 2002. SUMO: An open-source traffic simulation. In *Proc. 4th Middle East Symp. Simulation and Modelling (MESM2002)*.

Krauss, S. 1997. Microscopic modeling of traffic flow: Investigation of collision-free vehicle dynamics. Ph.D. thesis, Center for Parallel Computing.

Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S. P., and Fischer, S. 2005. Shawn: A new approach to simulating wireless sensor networks. In *Proceedings Design, Analysis, and Simulation of Distributed Systems (DASD05)*. 117–124.

Kwella, B. and Lehmann, H. 2000. Floating car data analysis of urban road networks. In *Proceedings Computer Aided Systems Theory - EUROCAST'99*, F. Pichler, R. Moreno-Díaz, and P. Kopacek, Eds. Lecture Notes in Computer Science, vol. 1798. Springer, Vienna, Austria.

Müller-Schloer, C., Schmeck, H., and Ungerer, T. 2004. Organic Computing – Proposal for a Focus Program.

Nagel, K. 1995. High-speed simulation of traffic flow. Ph.D. thesis, Center for Parallel Computing, Universität zu Köln, Germany.

Nagel, K. 2002. Cellular automata models for transportation applications. In *Proc. 5th Int. Conf. Cellular Automata for Research and Industry*. Springer LNCS, vol. 2493. 20–31.

Nagel, K. 2003. Traffic networks. In *Handbook of graphs and networks – From the genome to the internet*, S. Bornholdt and H. G. Schuster, Eds. Lecture Notes in Computer Science. Wiley-VCH, Berlin, Chapter 11.

Nagel, K. and Schreckenberg, M. 1992. A cellular automation model for freeway traffic. *Journal de Physique I France 2*, 2221–2229.

Nagel, K., Wagner, P., and Woesler, R. 2003. Still flowing: approaches to traffic flow and traffic jam modeling. *Operations Research 51,* 5, 681–710.

Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S., and Sheu, J.-P. 1999. The broadcast storm problem in a mobile ad hoc network. In *5th Annual ACM/IEEE Conf. MobiCom '99*. 151–162.

Oh, S., Kang, J., and Gruteser, M. 2006. Location-based flooding techniques for vehicular emergency messaging. In *2nd Int. Worksh. Vehicle-to-Vehicle Communications*. San Jose.

Organic Computing Initiative. 2004. A novel computing paradigm. `http://www.organic-computing.org`.

Perkins, C. E. 2001. *Ad hoc networking*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Pottmeier, A., Hafstein, S., Chrobok, R., Wahle, J., and Schreckenberg, M. 2003. The traffic state of the Autobahn network of North Rhine-Westphalia: An online traffic simulation. In *Pro. 10th World Cong. and Exh. on Intell. Transp. Syst. and Serv.* Doc. Nr. 2377.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. 2001. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM Press, 161–172.

Rickert, M. and Nagel, K. 1997. Experiences with a simplified microsimulation for the Dallas/Fort Worth area. *Int. J. Mod. Phys. C 8*, 133–153.

Rickert, M., Nagel, K., Schreckenberg, M., and Latour, A. 1996. Two-lane traffic simulation on cellular automata. *Physica A 231*, 534–550.

Schönhof, M. and Helbing, D. 2007. Empirical Features of Congested Traffic State and Their Implications for Traffic Modeling. *Transportation Science 41,* 2, 1–32.

Schreckenberg, M. and Selten, R. 2004. *Human behavior and traffic networks*. Springer-Verlag, Berlin.

Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., and Estrin, D. 2003. Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev. 33,* 1, 137–142.

Skutella, M. 2009. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization*, W. Cook, L. Lovász, and J. Vygen, Eds. Springer-Verlag, 451–482.

Surowiecki, J. 2004. *The wisdom of crowds*. Doubleday, London.

Treiber, M. and Helbing, D. 2001. Microsimulations of freeway traffic including control measures. *Automatisierungstechnik 49*, 478.

Wegener, A., Hellbrück, H., Fischer, S., Schmidt, C., and Fekete, S. 2007. AutoCast: An adaptive data dissemination protocol for traffic information systems. In *VTC 2007 Fall: 66th IEEE Vehicular Technology Conference*. Baltimore, USA. To appear.

Wegener, A., Schiller, E. M., Hellbrück, H., Fekete, S. P., and Fischer, S. 2006. Hovering data clouds: A decentralized and self-organizing information system. In *Proceedings of International Workshop on Self-Organizing Systems, Passau, Germany*.

Wischhof, L., Ebner, A., Rohling, H., Lott, M., and Halfmann, R. 2003. Adaptive broadcast for travel and traffic information distribution based on inter-vehicle communication. In *Proceedings of IEEE Intelligent Vehicles Symposium*. Columbus, Ohio, USA.

Xu, H. and Barth, M. 2006. An adaptive dissemination mechanism for inter-vehicle communication-based decentralized traffic information systems. In *Proceedings of IEEE Intelligent Transportation Systems*. 1207–1213.

Xu, Q., Mak, T., Ko, J., and Sengupta, R. 2004. Vehicle-to-vehicle safety messaging in DSRC. In *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. ACM Press, New York, NY, USA, 19–28.

Ye, F., Luo, H., Cheng, J., Lu, S., and Zhang, L. 2002. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the eighth annual international conference on Mobile computing and networking*. ACM Press, 148–159.

Appendix A

Algorithm 1: HDCs for traffic jams

```
initialization:
i = 0; etc. ;
buffer = ∅;
settimer(clock, smData);
onTimer(smData)
    q_before = Q_v;
    p_before = p_v;
    back = ∞;
    front = 0;
    P_v = 0; Q_v = 0;
    p_v = 0; q_v = 0;
    p = 0; q = 0;
    LBcast-CR(⟨smdata, ID, loc, v⟩);  //broadcast in CR
    settimer(next-multiple(t_data), Data);  //as Data follows smData
    s = 0;
onTimer(Data)
    t = clock;
    congestion_counter = 0;
    if  (congestion_participant = 1) ∨
        ((p = 1) ∧ (congestion_participant_before = 1)) ∨
        ((q = 1) ∧ (congestion_participant_before = 1)) then
        congestion_participant_before = 1;
    else
        congestion_participant = 0;
        p_before = 0;
        q_before = 0;
    if (status_back = active_back) ∧ (congestion_participant = 0) then
        status_back = idle;
        ahead = 0;
    if (status_front = active_front) ∧ (congestion_participant = 0) then
        status_front = idle;
    if (congestion_participant = 0 ∧ v ≤ CVmax) ∨
        (status_back = active_back) ∨
        (status_front = active_front) then
        LBcast(⟨data, ID, loc, v, p, q⟩);  //broadcast in R
        settimer(next-multiple(t_smdata), smData); //as smData follows Data
        congestion_participant_before = congestion_participant;
        congestion_participant = 0;
    else if (congestion_participant = 1) ∨
            (congestion_participant = 0 ∧ v > CVmax) then
        congestion_participant_before = congestion_participant;
        congestion_participant = 0;
        settimer(next-multiple(t_smdata), smData); //as smData follows Data
    i = 0;
LBrecv(m)
    buffer = buffer ∪ ⟨ m, clock⟩;
    settimer(clock + d, NewMessage);
```

```
Congestion
    if status_back = active_back then
        if (location_back < back) ∨ (location_back > back) then
            location_back = back;
        settimer(next-multiple(t_information), Information);
    if status_back = joining_back then
        status_back = active_back;
        settimer(next-multiple(t_information), Information);
        location_back = back;
    if status_back = idle then
        if q_before = 0 then
            location_back = back;
            state = v0_back;
            status_back = active_back;
            settimer(next-multiple(t_information), Information);
        else
            status_back = joining_back; //status to collect all dates

onTimer(Information)
    if status_back = active_back then
        LBcast(⟨Congestion, location_back, HDCfront_location, v ⟩);
        settimer(next-multiple(t_information), Information);
        if |loc-location_back| > r_HDC then
            status_back = idle; //If the distance is too big, the processor//
            becomes idle.  So, on the next call of Information no further
            //timer is set
            ahead = 0;

CongestionAhead
    if status_front = active_front then
        if (location_front < front) ∨ (location_front > front) then
            location_front = front;
        settimer(next-multiple(t_aheadInfo), AheadInfo);
    if status_front = joining_front then
        status_front = active_front;
        settimer(next-multiple(t_aheadInfo), AheadInfo);
        location_front = front;
    if status_front = idle then
        if p_before = 0 then
            location_front = em front;
            state = v0_front;
            status_front = active_front;
            settimer(next-multiple(t_aheadInfo), AheadInfo);
        else
            status_front = joining_front;

onTimer(AheadInfo)
    if status_front = active_front then
        LBcast(⟨hdcdistance, location_front⟩);
    if |loc-location_front| > r_HDC then
        status_front = idle;
```

```
onTimer(NewMessage)
  let m = min(m:⟨ m, t ⟩ ∈ buffer, t = clock - d);
  if m = ⟨data, ident, l, g, p-value, q-value⟩) then
    //checking critical values
    env_i = ⟨ident, l, g⟩;
    if (| loc - l | < CR) ∧ (v < CV) then
      back = min{back, loc, l};
      if back==loc then back_id = ID;
      if back==l then back_id = env_i.ident;
      P_v = p-value of back_id; Q_v = q-value of back_id;
      front = max{front, loc, l };
      if front==loc then front_id = ID;
      if front==l then front_id = env_i.ident;
      p_v = p-value of front_id; q_v = q-value of front_id;
      congestion_counter++; s = 1;
      congestion_participant = 1;
    //comparison:  transmitting processor - already received messages
    for j, 1, i − 1 do
      if (| l - env_j.l | < CR) ∧ (g < CV) then
        back = min{back, l, env_j.l};
        if back==env_j.l then back_id = env_l.ident;
        if back==l then back_id = env_i.ident;
        P_v = p-value of back_id; Q_v = q-value of back_id;
        front = max{front, l, env_j.l };
        if front==env_j.l then front_id = env_j.ident;
        if front==l then front_id = env_i.ident;
        p_v = p-value of front_id; q_v = q-value of front_id;
        if s = 0 then
          congestion_counter++;
          s = 1;
  if (congestion_counter > 0) ∧ (|back - loc| < r_HDC) ∧ (P_v = 0)
     ∧ (clock ≥t + t_ab + d) then
    invoke Congestion;
    congestion_participant = 1;
  if (congestion_counter > 0) ∧ (|front - loc| < r_HDC) ∧ (q_v = 0)
     ∧ (clock ≥ t + t_ab + d) then
    invoke CongestionAhead;
    congestion_participant = 1;
  if ahead = 0 then HDCfront_location = front; i++;
  if m = ⟨Congestion, l, location_hdc_front, g⟩ then
    if l > loc then LBcast(⟨Congestion, l, location_hdc_front, g⟩);
    if |l - loc| < r_HDC then status_back = joining_back;
  if m = ⟨hdcdistance, L_front⟩ then
    if status_back = active_back then
      HDCfront_location = L_front; ahead = 1;
    else if congestion_participant = 1 then
      LBcast(⟨hdcdistance, L_front⟩);
  if m = ⟨smData, ident, l, g⟩ then
    if l < loc then p = 1; //right neighbor
    if l > loc then q = 1; //left neighbor
```